Date=16/11/2020
Lecture By=Manish Mahant
Subject ⇒React Events

IN PREVIOUS LECTURE **(QUICK RECAP) Date-13/11/2020**	**In Today's Lecture (Overview)**

React Events

Just like HTML, React can perform actions based on user events.

React has the same events as HTML: click, change, mouseover etc.

Adding Events

React events are written in camelCase syntax:

`onClick` instead of `onclick`.

React event handlers are written inside curly braces:

`onClick={shoot}` instead of `onClick="shoot()"`.

React:

```
<button onClick={shoot}>Take the Shot!</button>
```

HTML:

```
<button onclick="shoot()">Take the Shot!</button>
```

Event Handlers

A good practice is to put the event handler as a method in the component class:

Example:

Put the `shoot` function inside the `Football` component:

```
class Football extends React.Component {
  shoot() {
    alert("Great Shot!");
  }
  render() {
    return (
      <button onClick={this.shoot}>Take the shot!</button>
    );
  }
}

ReactDOM.render(<Football />, document.getElementById('root'));
```

Bind `this`

For methods in React, the `this` keyword should represent the component that owns the method.

That is why you should use arrow functions. With arrow functions, `this` will always represent the object that defined the arrow function.

Example:

```
class Football extends React.Component {
  shoot = () => {
    alert(this);
    /*
    The 'this' keyword refers to the component object
    */
  }
  render() {
    return (
      <button onClick={this.shoot}>Take the shot!</button>
    );
  }
}

ReactDOM.render(<Football />, document.getElementById('root'));
```

Why Arrow Functions?

In class components, the `this` keyword is not defined by default, so with regular functions the `this` keyword represents the object that called the method, which can be the global window object, a HTML button, or whatever.

Read more about binding `this` in our React ES6 'What About this?' chapter.

If you *must* use regular functions instead of arrow functions you have to bind `this` to the component instance using the `bind()` method:

Example:

Make `this` available in the `shoot` function by binding it in the `constructor` function:

```
class Football extends React.Component {
  constructor(props) {
```

```
    super(props)
    this.shoot = this.shoot.bind(this)
  }
  shoot() {
    alert(this);
    /*
    Thanks to the binding in the constructor function,
    the 'this' keyword now refers to the component object
    */
  }
  render() {
    return (
      <button onClick={this.shoot}>Take the shot!</button>
    );
  }
}

ReactDOM.render(<Football />, document.getElementById('root'));
```

Passing Arguments

If you want to send parameters into an event handler, you have two options:

1. Make an anonymous arrow function:

Example:

Send "Goal" as a parameter to the `shoot` function, using arrow function:

```
class Football extends React.Component {
  shoot = (a) => {
    alert(a);
  }
  render() {
    return (
      <button onClick={() => this.shoot("Goal")}>Take the shot!</button>
    );
  }
}
```

```
ReactDOM.render(<Football />, document.getElementById('root'));
```

Or:

2. Bind the event handler to `this`.

Note that the first argument has to be `this`.

Example:

Send "Goal" as a parameter to the `shoot` function:

```
class Football extends React.Component {
  shoot(a) {
    alert(a);
  }
  render() {
    return (
      <button onClick={this.shoot.bind(this, "Goal")}>Take the
shot!</button>
    );
  }
}

ReactDOM.render(<Football />, document.getElementById('root'));
```

Questions For the Self-Practice

CC
https://au-assignment.s3.ap-south-1.amazonaws.com/Week_21_Day_1_Challenge-13cc7080-d63c-40f1-b8e0-95611955486a.pdf

Assignment
https://au-assignment.s3.ap-south-1.amazonaws.com/Week_21_Day_1_Assignment-0540d5ad-313d-4f07-afc2-2c024cf60ab7.pdf